

REMARKS

Claims 1-61 are pending. In the Office Action dated September 28, 2004, the Examiner took the following action: (1) rejected claims 25-27, 29, 32-36, 38-40, 46-50, 52-54 and 60-61 under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent No. 6,115,809 to Mattson, Jr. *et al.*; (2) rejected claims 1-4, 6-8, 13-15 and 17-24 under 35 U.S.C. § 103(a) as being obvious over the patent to Mattson, Jr. *et al.* and in view of U.S. Patent No. 5,765,037 to Morrison, *et al.*; (3) rejected claims 5, 12 and 16 under 35 U.S.C. § 103(a) as being obvious over the patent to Mattson, *et al.* and Morrison, *et al.* as applied to claims 1 and 15 above, and further in view of U.S. Patent No. 6,272,599 to Prasanna; (4) rejected claims 9-11 under 35 U.S.C. § 103(a) as being obvious over the patent to Mattson, *et al.* and Morrison, *et al.* as applied to claim 1 above, and in further view of U.S. Published Patent Application No. 2002/0078268 to Lasserre; (5) rejected claims 28, 37, 45 and 51 under 35 U.S.C. § 103(a) as being obvious over the patent to Mattson, *et al.* as applied to claims 25, 34 and 48 above, in view of Prasanna; (6) rejected claims 30-31, 41-44 and 55-58 under 35 U.S.C. § 103(a) as being obvious over the patent to Mattson, *et al.* as applied to claims 25, 34 and 48 above, in view of Lasserre; and (7) rejected claim 59 under 35 U.S.C. § 103(a) as being obvious over the patent to Mattson, *et al.* and Lasserre as applied to claim 58 above, and further in view of Prasanna.

The disclosed embodiments of the invention will now be discussed in comparison to the applied references. Of course, the discussion of the disclosed embodiments, and the discussion of the differences between the disclosed embodiments and the subject matter described in the applied references, do not define the scope or interpretation of any of the claims. Instead, such discussed differences merely help the Examiner appreciate important claim distinctions discussed thereafter.

The disclosed invention is directed to a system and method for improving the efficiency of caching instructions executed by a processor. Rather than relying on the conventional approach of caching instructions based on either how frequently or how recently an instruction was executed, the disclosed caching system and method determines the cacheability of an instruction at compilation based on other factors, some of which are described in the

specification. Based on this cacheability determination during compilation, the instructions are marked with at least one bit corresponding to the determination. During execution of the instructions, the cacheability bit is detected and used by a computer system to either cache or not cache the instruction.

For reasons that will be explained below, it is important to understand the difference between caching, branch prediction and instruction reordering, all of which are techniques to improve the rate at which a processor can execute instructions. Caching attempts to store instructions that are frequently executed in a cache memory. The processor can fetch instructions from cache memory much more quickly than the processor can fetch instructions from main or system memory. As a result, the instructions that are frequently executed can be fetched by the processor much more quickly than other instructions. Cache memory generally uses static random access ("SRAM") devices, which is relatively expensive, *i.e.*, has a relatively high price per byte of storage. In contrast, main or system memory, which generally uses dynamic random access ("DRAM") devices, is relatively inexpensive, *i.e.*, has a relatively low price per byte of storage. It is therefore not feasible to store all of the instructions that will be executed in the cache memory. Instead, the instructions that are less likely frequently executed are stored in main memory. As a result, in a cache system, a choice is made as to which instructions are stored in cache memory and which instructions are stored in main memory or some other memory.

It is important to understand that a cache system is distinctly different from a branch prediction system. Branch prediction systems are used to prefetch instructions in a pipelined processing system in which instructions are fetched from either cache memory or main memory before the instructions are to be executed. These prefetched instructions are then stored in a processing queue in a small register in the processor so that they are immediately available to the processor when they are to be executed. Unfortunately, the efficiency of prefetching instructions can be compromised when a branch instruction is encountered because it may not be possible to determine which branch will be taken before the branch instruction is executed. Without knowing which branch will be taken, it is not possible to know which instructions should be prefetched, *i.e.*, the instructions in branch 1 or the instructions in branch 2. Branch prediction techniques have been developed to provide an educated guess about which branch will

be taken based on such factors as which branch was most recently taken or which branch was most frequently taken. The instructions in the predicted branch can then be prefetched so that they will be immediately available to the processor when it is time to execute them if the branch prediction was correct. If the branch prediction was incorrect, the queue of prefetched instructions must be flushed, i.e., erased, and processing stalls while the instructions in the correct branch are fetched.

Some of the common differences between a cache system and a branch prediction system can be summarized as follows:

- In a caching system, a choice is made about which instructions are to be stored in the cache memory and which instructions are to be stored in main memory or some other memory; in a branch prediction system, all of the instructions in the predicted branch are generally stored in the same register or memory
- Branch prediction systems are used only when branch instructions are involved; cache systems are used to store instructions other than branch instructions, although they may also store branch instructions
- Branch prediction systems are used only in pipelined processing systems in which instructions are prefetched; cache systems can be used in both processing systems where prefetching does not occur and processing systems where prefetching does occur

A caching system is also quite different from instruction re-ordering techniques in which the instructions in a program are re-arranged so that they can be processed most efficiently. In instruction re-ordering, a program is analyzed to determine the manner in which the instructions will be executed, and the relative order of the instructions can be changed to allow the program to be executed more efficiently. For example, if an instruction repetitively calls a subroutine that is located in a different area of the program, the execution efficiency may be improved by relocating the subroutine closer to the calling instruction.

In applicant's disclosed method and system, the efficiency at which a program can be executed is improved by more efficiently caching instructions based on factors determined during compilation. The instructions are then marked accordingly and used by the

processor to determine whether an instruction should be stored in cache memory or main memory. In contrast to branch prediction, the marked instruction may be an instruction other than a branch instruction, and the caching system may be used with a processor that does not prefetch instructions.

In the prior Office Actions, the claims were finally rejected on the basis of the Serocki *et al.* patent, which disclosed inserting branch target address hints into a program during compilation. The hints identify the branches that the program is expected to take during execution. The program and hints are then stored in cache memory so the hints can be used by the processor to predict which branch will be taken during execution. In response to applicant's remarks, the Examiner has recognized that applicant's disclosed caching method and system was different from branch prediction systems as disclosed in the Serocki *et al.* patent. The Examiner therefore withdrew the final rejection based on the Serocki *et al.* patent, and applicant wishes to thank the Examiner for her willingness to reconsider this rejection. Unfortunately, the Mattson, Jr. *et al.* patent, which has been substituted for the Serocki *et al.* patent, also discloses a branch prediction system rather than a caching system.

The Mattson, Jr. *et al.* patent discloses branch prediction system in which a program of instructions are first traced during execution so the branching behavior of the instructions can be determined. (*See, e.g.*, "The present invention first profiles branch instructions within a trace to record branching behavior." Abstract). The branch instructions detected in the trace are divided into two groups. A first group consists of branch instructions having "strong" branching behavior in which the same branch was taken at least 80% of the time. A second group consists of branch instructions having "weak" branching behavior in which the same branch was taken less than 60% of the time. [Abstract]. "Branch instructions that are profiled to have 'strong' branching behavior...are placed in the group of branch instructions that are statically predicted. Branch instructions that are provided to have 'weak' branching behavior...are placed in the group of branch instructions that are dynamically predicted." [Abstract]. The two different groups of branch instructions are therefore processed using different algorithms.

Unfortunately, the two different groups of branch instructions are referred to in the Mattson, Jr. *et al.* patent as being placed in two different "caches," namely a "static code

cache” and a “dynamic code cache.” However, the use of the term “cache” in this manner has nothing to do with a caching system as described by applicant. In fact, as will be demonstrated below, both the “static code cache” and the “dynamic code cache” are stored in main memory, albeit in different pages of the main memory, and they have prediction flags set to different values. The Mattson, Jr. *et al.* patent does not describe or suggest storing any part of either the “static code cache” or the “dynamic code cache” in a cache memory. Neither does the Mattson, Jr. *et al.* patent describe or suggest storing some of the branch instructions in cache and others of the branch instructions in main memory. Instead, all branch instructions are either in the “static code cache” or the “dynamic code cache.”

The fact that the “static code cache” and the “dynamic code cache” are simply different pages in main memory is apparent from the teachings in a number of locations in the Mattson, Jr. *et al.* patent. For example, the cover which is a copy of Figure 5, shows the static code cache 38 and the dynamic code cache 40 as being separate “physical memory pages.” As the Examiner may be aware, the term “pages” is typically used to refer to different rows or other contiguous locations of main memory. Similarly, the Mattson, Jr. *et al.* patent states between line 55 of column 5 and line 3 of column 8:

Prediction strategy flags 42 and 44 represent the branch prediction flags associated with the memory pages that comprise static code cache 38 and dynamic code cache 40, respectively. As shown in FIG. 2, certain prior art computer systems manufactured by the Hewlett-Packard Company include a branch prediction strategy flag in each translation lookaside buffer (TLB). The branch prediction strategy flag in each TLB determines the branch prediction strategy of branch instructions residing in the physical memory pages associated with the TLB. Accordingly, in such a computer system, all the branch prediction strategy flags in all the TLBs associated with the physical memory pages that comprise static code cache 38 are set to “static”, and all branch prediction strategy flags in all the TLBs associated with the physical memory pages that comprise dynamic code cache 40 are set to “dynamic”.

What is described in this excerpt is simply storing the strong and weak branch instructions in different pages of main memory (*i.e.*, “physical memory”), which are identified by prediction flags set to “static” or “dynamic,” respectively, in translation lookaside buffers associated with the different pages of main memory. While the patent does not specifically state that the “physical memory” is main memory, neither does the patent state that the “physical memory” is a cache memory as that term is used by applicant. As the Examiner is undoubtedly aware, a patent is a reference only for what it clearly teaches. It is not a reference for what it might teach if it were not ambiguous. Furthermore, it is apparent from the context in which the physical memory is described in the Mattson, Jr. *et al.* patent that the physical memory is, in fact, main memory rather than cache memory as that term is used by applicant. For example, if the physical memory was cache memory rather than main memory, then there would be some determination made as to which instructions were cached and which instructions were stored in main memory.

The patent to Mattson, Jr. *et al.* has been cited in combination with the patent to Morrison, *et al.* The Morrison *et al.* patent describes a technique for more efficiently executing branched instructions in a parallel computer system by re-ordering the instructions. The Morrison *et al.* patent teaches analyzing the instructions in a program during compilation based on the manner in which the instruction uses resources, such as whether instructions are independent and can thus be executed in parallel. Each instruction is then assigned an execution time at step 130, *i.e.*, the amount of time required to execute the instruction. This execution time is referred to as the “instruction firing time” or “IFT.” The instruction is also assigned other data at step 130 such as the identity of one of several processors that will execute the instruction. Finally, execution sets are built at step 140 by a re-ordering of the instructions based upon the instruction firing time, the identity of the processor executing the instruction, etc. Therefore, the patent to Morrison, *et al.* does not supply the teachings that are missing from the teachings of the Morrison *et al.* patent.

The remaining references cited in the Office Action likewise fail to suggest the teachings of applicant’s disclosed method and system that are missing from the teachings of the Mattson, Jr. *et al.* patent and the teachings of the Morrison *et al.* patent.

Turning, now, to the claims, claim 1, which has been rejected as being obvious over the Mattson, Jr. *et al.* patent in view of the Morrison *et al.* patent, specifies a computer system having cache circuitry, a main memory and a processor controlled by a computer program. Both the cache memory and the main memory store information related to a computer program. The processor directs selected portions of the information to the cache circuitry “based at least in part on cacheability determinations made during compilation of the computer program.” As explained above, neither the Mattson, Jr. *et al.* patent nor the Morrison *et al.* patent describe storing information related to a computer program in main memory and then making cacheability determinations about that information to direct selected portions of the information to the cache circuitry. Claim 1 is thus clearly patentable over the cited references.

Independent claim 25, which is directed to a system for determining which portions of a program code to cache and which not to cache, has been rejected as being anticipated by the Mattson, Jr. *et al.* patent. The claimed system includes a processor connected to a memory device containing a program code. The processor is controlled by the program code “to direct selected portions of the program code to a cache based at least in part on cacheability determinations made during compilation of a computer program.” As explained above, the Mattson, Jr. *et al.* patent is directed to branch prediction determinations; it does not describe or suggest making cacheability determinations during compilation or using the cacheability determinations to control caching during execution of the instructions. Nor does the Mattson, Jr. *et al.* describe directed “selected portions” of program code to a cache.

Independent method claim 34 is directed to a method of controlling the cacheability of information in a computer system. This claim was also rejected as being anticipated by the Mattson, Jr. *et al.* patent. However, it should be readily apparent that the Mattson, Jr. *et al.* patent does not describe making cacheability determinations for information associated with a computer program that is being compiled, nor does it describe marking at least selected portions of information according to the cacheability determinations. Instead, it describes marking branch instructions according to their branching behavior, *i.e.*, strong or weak. Likewise, the Mattson, Jr. *et al.* patent fails to teach detecting markings of selected portions of information during execution and directing the selected portions of information to the cache

circuitry according to the markings. Instead, both strong and weak instructions are directed to different pages of the same physical memory.

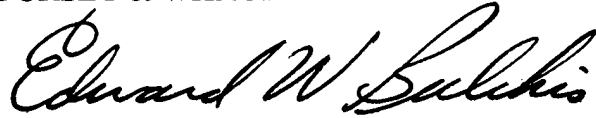
The final Independent claim is claim 48, which is directed to a method for compiling a computer program. The claimed method includes making cacheability determinations for information associated with a computer program and then marking at least selected portions of the information according to the determinations. Claim 48 was also rejected as being anticipated by the Mattson, Jr. *et al.* patent. However, as explained above, the Mattson, Jr. *et al.* patent does not teach making cacheability determinations nor does it teach marking at least selected portions of the information according to the determinations. Instead, it teaches making branching behavior determinations and marking the computer program according to the branching behavior determinations so that the branch instructions can be processed by either static or dynamic procedures.

The remaining claims in the application patentably distinguish over the cited references because of their dependency on patentable independent claims and because of the additional limitations added by those claims.

All of the claims in the application, *i.e.*, claims 1-61, are clearly allowable.
Favorable consideration and a timely Notice of Allowance are therefore earnestly solicited.

Respectfully submitted,

DORSEY & WHITNEY LLP



Edward W. Bulchis
Registration No. 26,847
Telephone No. (206) 903-8785

EWB:dms

Enclosures:

Postcard

Fee Transmittal Sheet (+copy)

DORSEY & WHITNEY LLP
1420 Fifth Avenue, Suite 3400
Seattle, WA 98101-4010
(206) 903-8800 (telephone)
(206) 903-8820 (fax)

h:\ip\documents\clients\micron technology\00\500050.01\500050.01 amend oa 092804.doc